Abstract

Embedded processors are becoming increasingly popular and widely used in several systems. Embedded systems are extremely sensitive to cost and power consumption while still requiring performance. In this thesis, we consider two aspects of compilation techniques for embedded systems, one that relates to cost, viz., code size reduction, and the other energy reduction.

DSP processors have address generation units that can perform address computation in parallel with other operations. This feature reduces explicit address arithmetic instructions, often required to access locations in the stack frame, through auto-increment and decrement addressing modes, thereby decreasing the code size. Effective utilization of auto-increment and decrement modes requires an intelligent placement of variables in the stack frame which is termed as "offset assignment". Although a number of algorithms for efficient offset assignment have been proposed in the literature, they do not consider possible instruction reordering to reduce the number of address arithmetic instructions. In this thesis, we propose an unified approach which combines instruction reordering and algebraic transformations to reduce the number of address arithmetic instructions. The proposed approach has been implemented in the SUIF compiler framework. We conducted our experiments on a set of real programs and compared our approach with four existing approaches.

Most embedded devices typically contain several DRAM chips (multiple memory modules) which can be individually put in low power mode when a memory module is not in use. Previous work on compiler optimizations for architectures involving DRAM chips do not consider the effect of program transformations on the execution time, and their

impact on energy consumption. In this thesis, we study the effect of loop and code transformations for such architectures in the presence of a cache. We propose a *weighted fission* heuristic which decides between loop fission and loop fusion between a pair of statements based on the priorities given to performance and power. We also propose an array layout heuristic to allocate arrays to memory modules/banks such that more memory banks can be put into low-power mode for longer duration. The weighted fission and the array allocation heuristics have been implemented in the SUIF compiler framework and an energy simulator is implemented in the Simplescalar tool set. We obtained our performance and power results by running experiments on a set of array-intensive benchmarks. Our results indicate that loop fission and array layout heuristics in general facilitate putting the memory modules in low-power mode for longer duration. However, this does not necessarily result in a corresponding decrease in energy consumption due to an increase in the number of cache misses or an increase in loop overheads, which, in turn, increases the execution time of the application causing an increase in the energy consumption.